# The Guessing App

Thesis by Carl Schlyter

for Medieinstitutet

”Webbutvecklare inom e-handel”

2020-04-16

# Documentation for the application

Content

[The Guessing App 1](#_Toc37942552)

[Documentation for the application 1](#_Toc37942553)

[1. The Guessing App – the idea 1](#_Toc37942554)

[1.1 The idea 1](#_Toc37942555)

[2. The application’s parts 1](#_Toc37942556)

[2.1 The database 1](#_Toc37942557)

[2.2 The API 2](#_Toc37942558)

[2.3 Front end 2](#_Toc37942559)

## The Guessing App – the idea

1.1 The idea  
The idea with the The Guessing App is to make it possible for advertizing agencies to activate their customers through a guessing competition. The Guessing App will then be adapted each time for each customer with relevant layout, logos etc and it will be decided upon, with the customer, what the guessing would be about. It could be anything, be it the coming weather, the finalists in the Eurovision Song Contest, the number of medals in a coming Olympic Games or, like in this example, the result in football matches in a football championship.

## The application’s parts

2.1 The database

The database in MySQL consists in the example of four tables: Users, Betters, Matches and Bets. All tables have automatic primary-key-id-fields (UUID) and the password in the Users table is hashed. I have, in order to scale down this thesis, chosen not to use the password or a login and it could be argued that that is quite alright with this kind of service.

I have been using MySQL-Triggers in the Users- and Matches tables respectively, to automatically update other tables given certain events:

**Users –** After ”insert”, new rows are automatically being created in the Betters table where the user is anonymized.

**Matches** – After ”update”, the Bets table is automatically updated with actual match result and with points with respect to the relationship between the match result and the registered guess in the Bets table.

The Matches table is supposed to be updated by an administrator, but there is no such interface right now. Instead the updating will have to be made by the administrator directly in the database.

2.2 The API

The active models/endpoints of the API in the example are:

**Game/read –** To display a list of upcoming matches with match date later than today.

**User/create** – To create a user in the database.

**User/read** – To display a list of users with their points from already finished matches.

**Bet/read** – To display a list it averages guesses by guessers on upcoming matches.

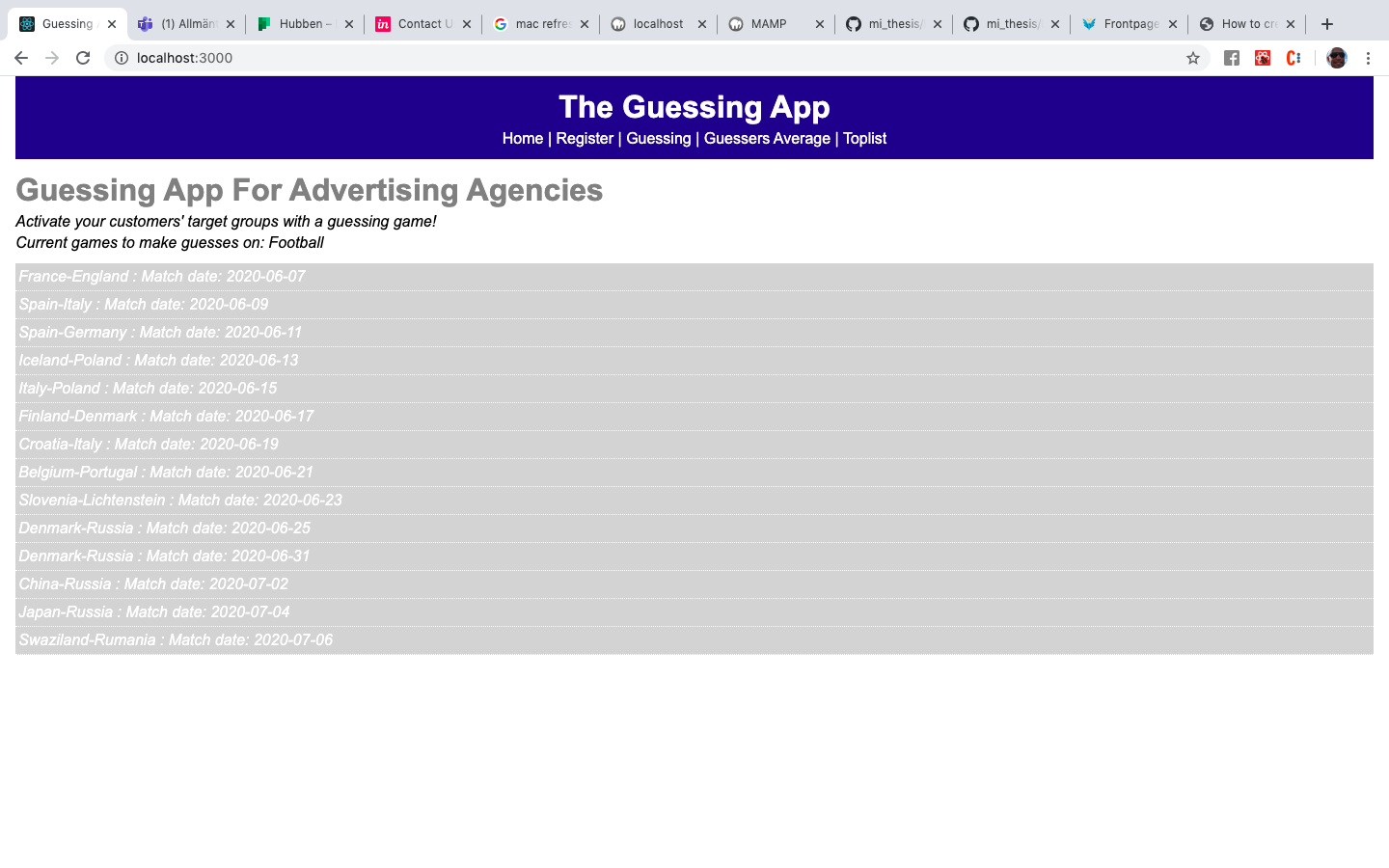
**Bet/create** – To create a guess.

2.3 Front end

The front end part is developed in REACT with Node Package Manager / Create React App and the below interface and functions:

***2.3.1 The ”Home” page***

The start page with a list of current matches to guess the result on:



***2.3.2 The ”Register” page***

With a form for registering a user::

A screenshot of a social media post

Description automatically generated

***2.3.3 The ”Guessing” page***

With a form for registering a guess:

**A screenshot of a social media post

Description automatically generated**

***2.3.4 The ”Guessers Average” page***

With a list of upcoming matches where someone has guessed the result and the average guesses on those matches:

**A screenshot of a social media post

Description automatically generated**

***2.3.5 The ”Toplist” page***

With a list of users who have guessed the result on already finished matches ranked by total points.

A screenshot of a social media post

Description automatically generated

\*\*\*\*\*\*\*